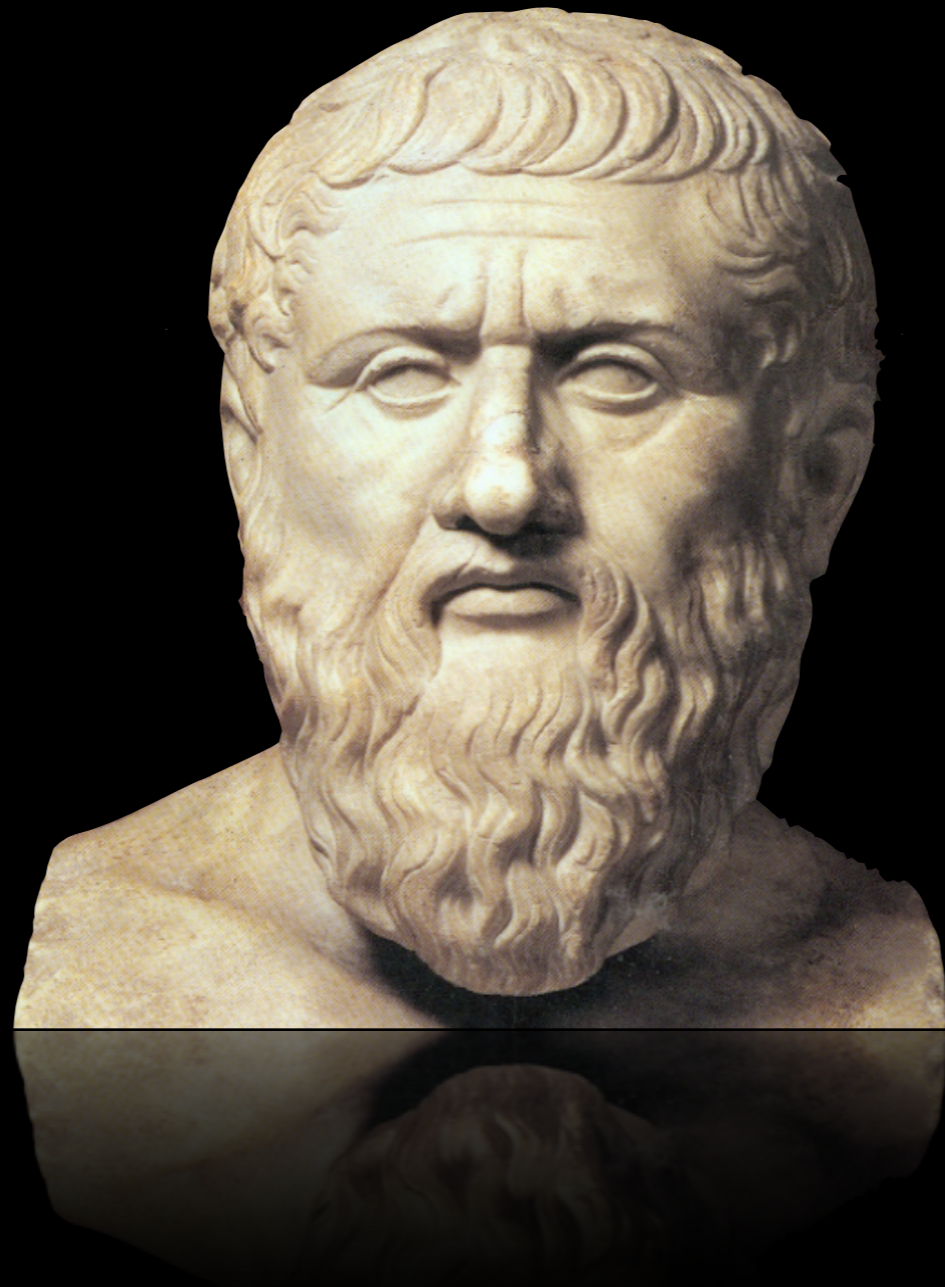


What makes
code
beautiful?

Marcel Molina

<http://marcelmolina.com>

Plato

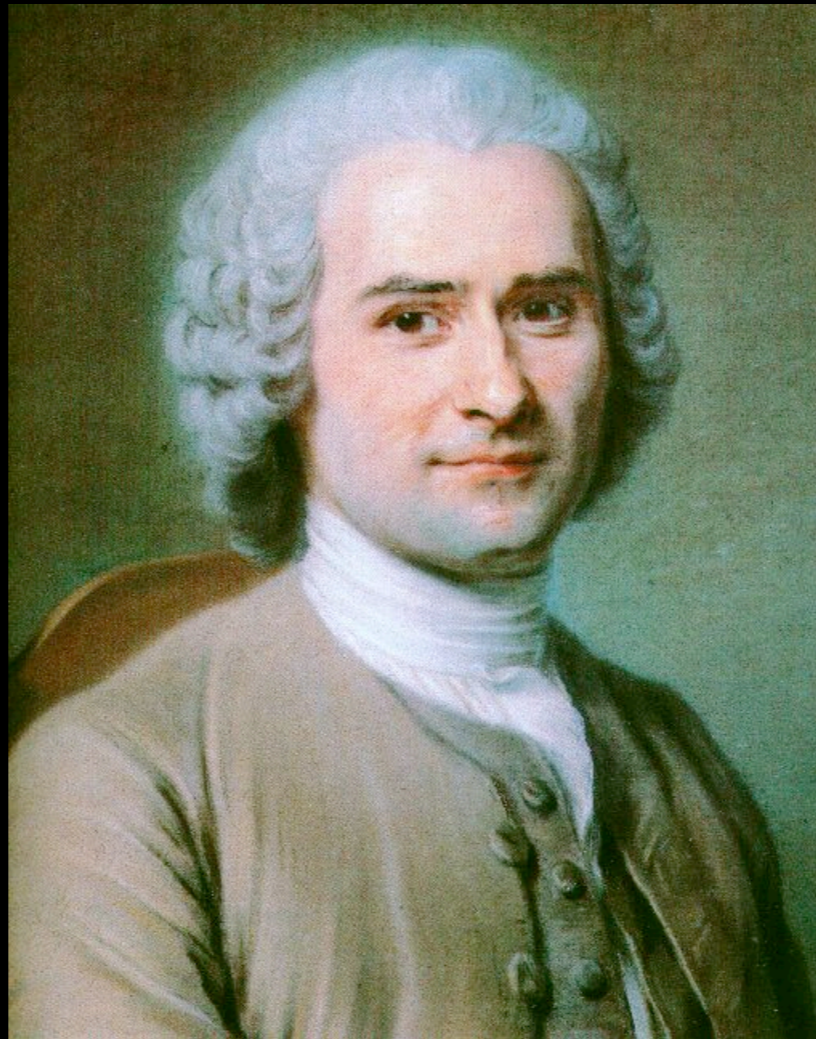


“The ability to grasp mere appearances cannot lead to adequate understanding.”

“At its worst, the appreciation of beauty
can **mire us in the world of sense
experience...**

**...but at its best it can lead to
the understanding of goodness.’**

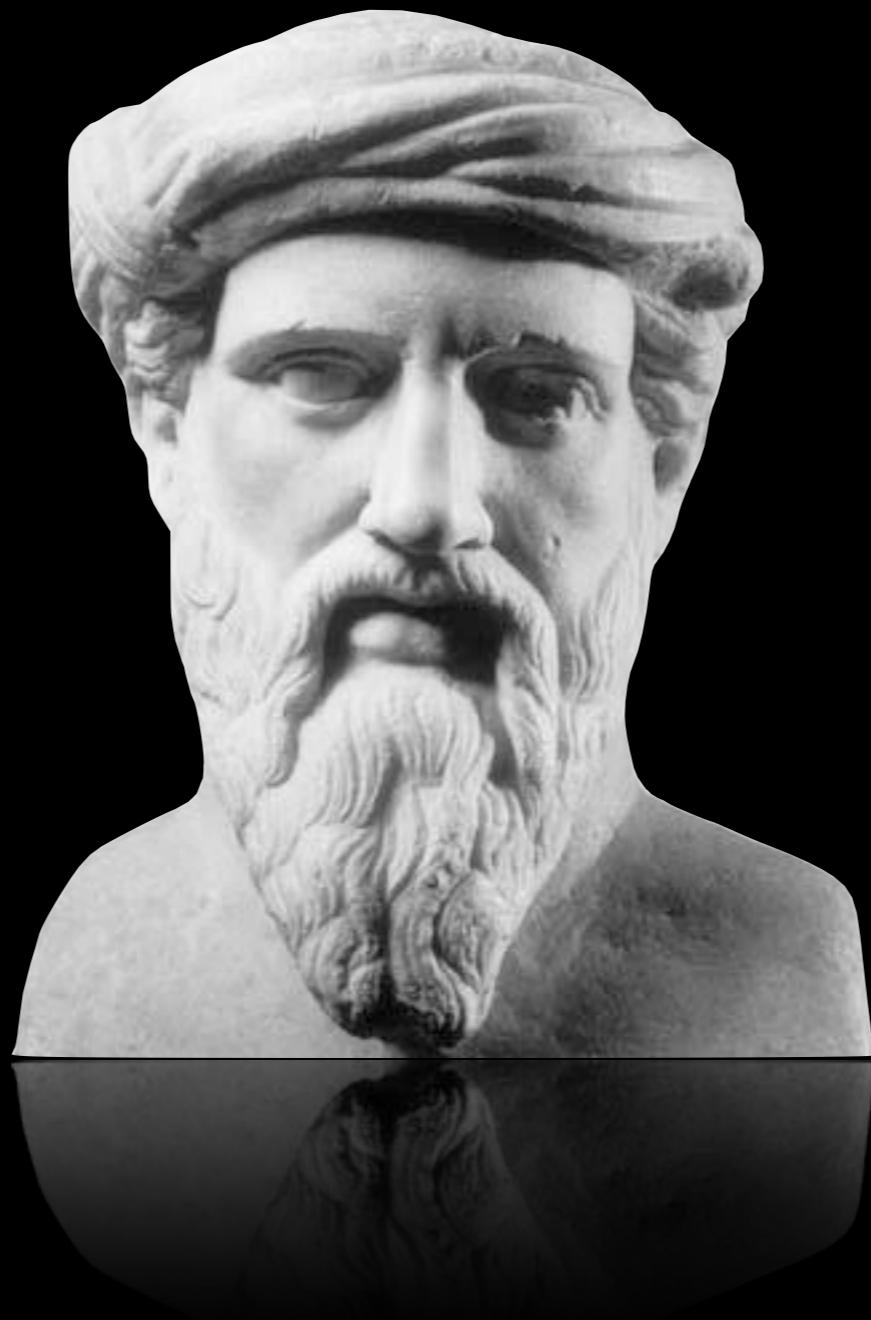
Jean-Jacques Rousseau



“I have always believed that **good** is none
other than **Beauty in action...**

...and both have a common source in
well-ordered nature.”

Pythagoras



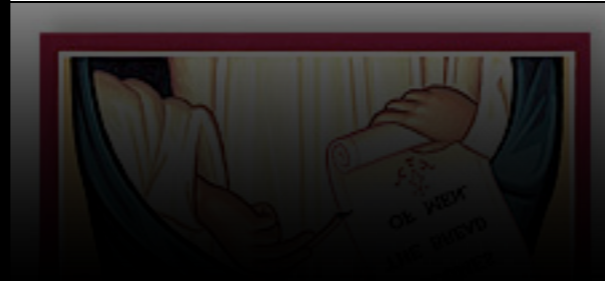
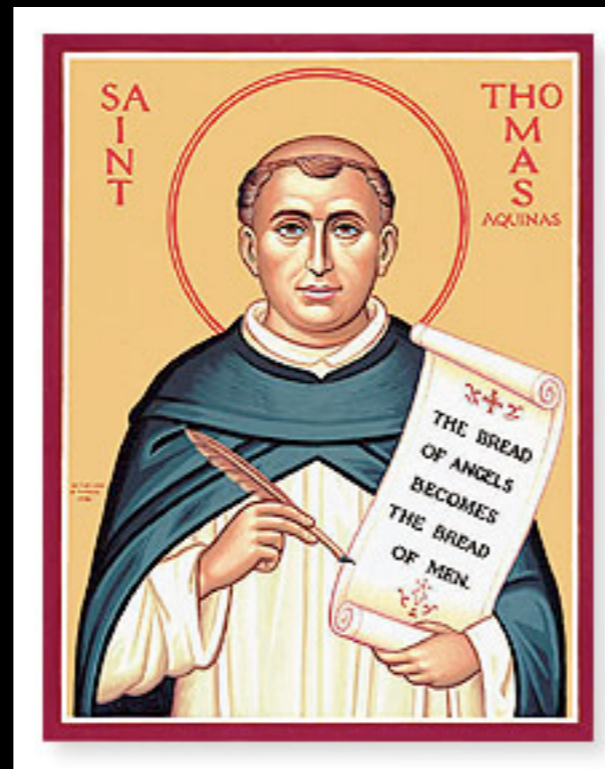
“All things exist because they are **ordered**...

...and they are ordered because they are the
realization of mathematical laws.’

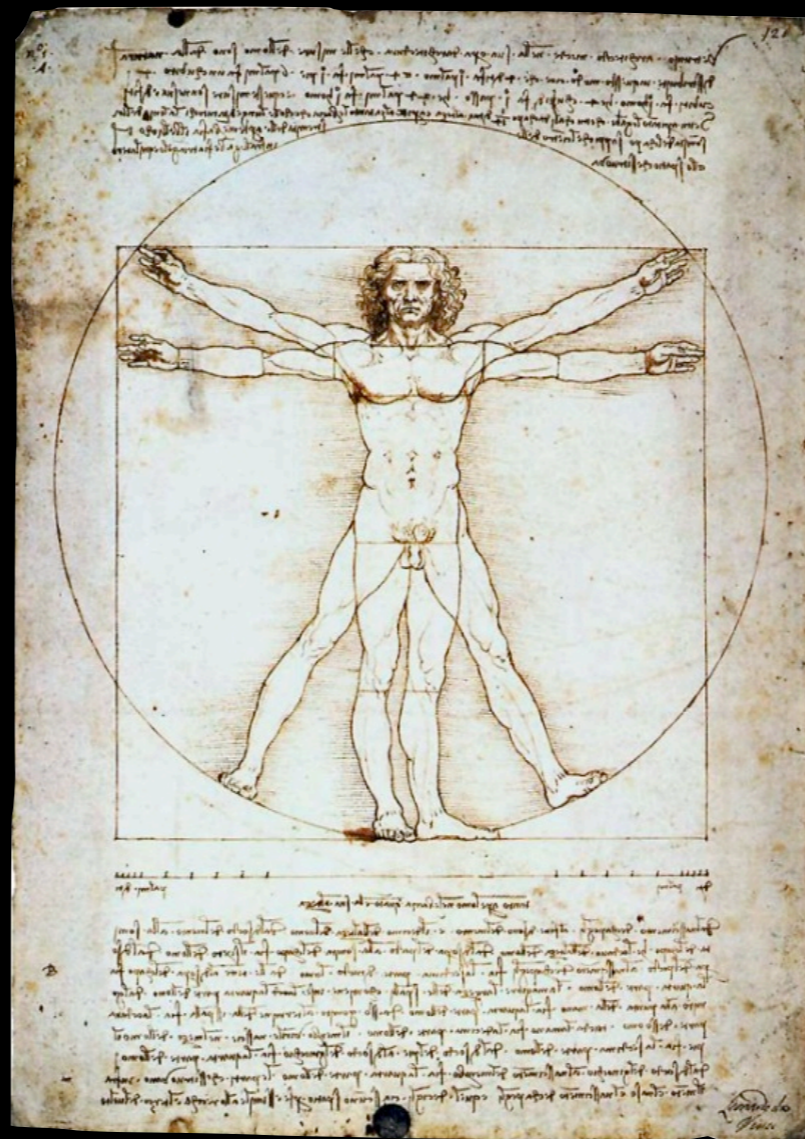




Thomas Aquinas



Proportion



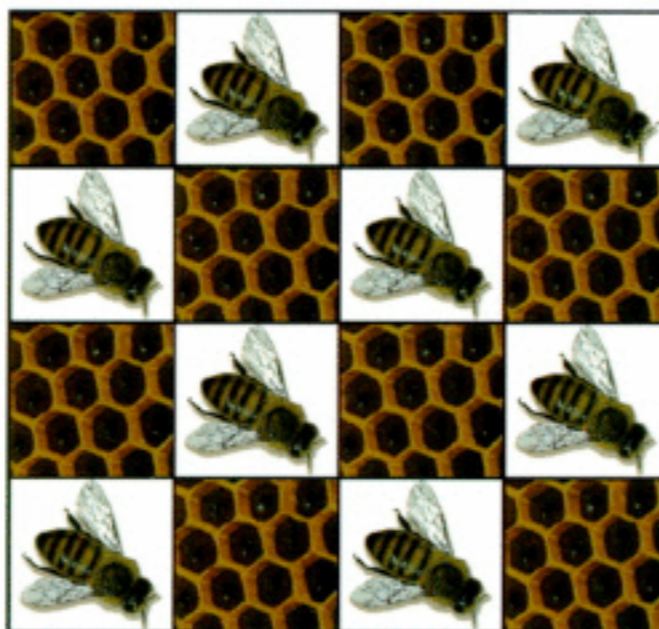
Integrity



Clarity



SMALLTALK BEST PRACTICE PATTERNS



KENT BECK

KENT BECK

“This pattern emphasizes
readability.”

“One line methods are there to
communicate.”

“Objects are manageable if they are
chopped into little pieces.”

“Most good Smalltalk methods fit into a few lines.”

“You **don't spend three or four lines** expressing iteration,
you **spend one word.**”

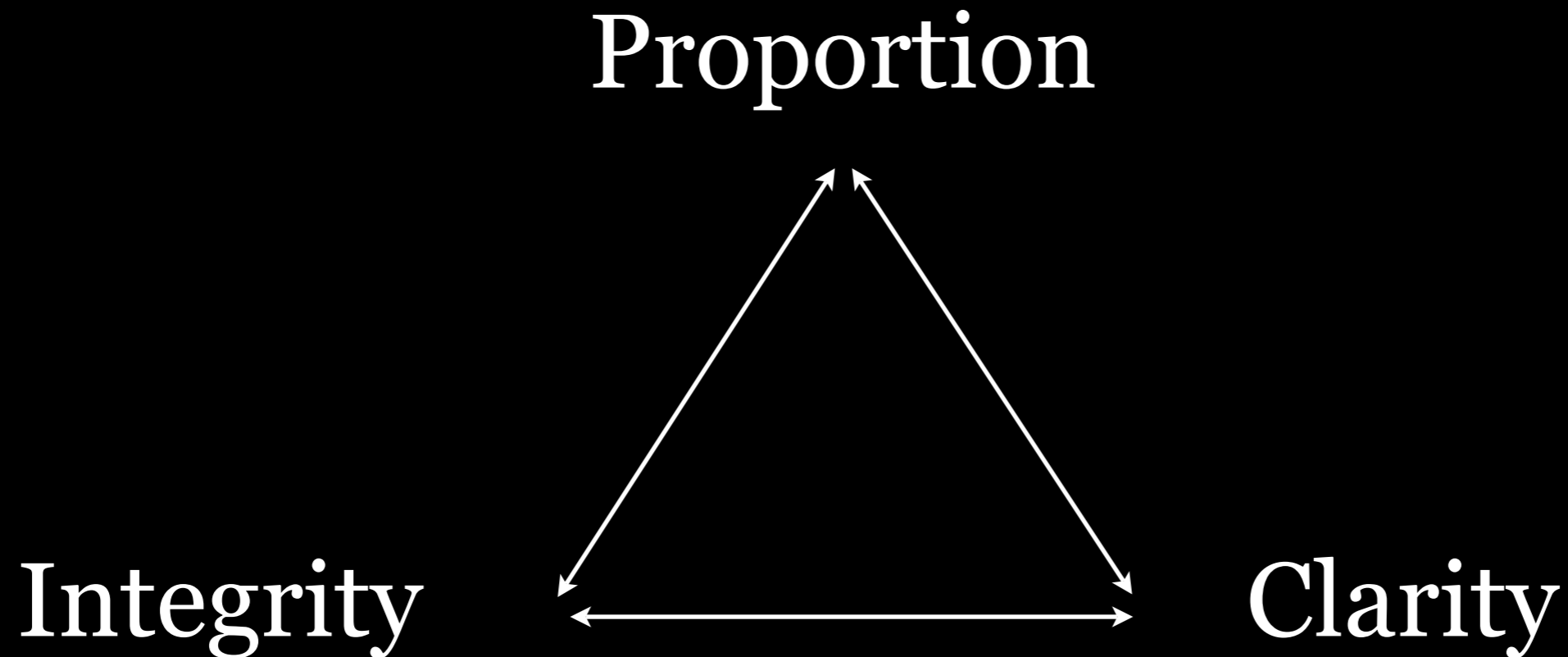
“The problem with code like this is
you can't read it and understand what is going on.”

AddNumbers:

```
std          ; go from LSB to MSB
clc          ;
pushf       ; save carry flag
.top
mov ax,0f0fh ; convert from ASCII BCD to BCD
and al,[si]  ; get next digit of number2 in al
and ah,[di]  ; get next digit of number1 in ah
popf        ; recall carry flag
adc al,ah    ; add these digits
aaa         ; convert to BCD
pushf       ;
add al,'0'   ; convert back to ASCII BCD digit
stosb       ; save it and increment both counters
dec si      ;
loop .top    ; keep going until we've got them all
popf        ; recall carry flag
ret         ;
```

```
16f      ?
bobt     ? 16c9tj c9llλ 4t98
toob    *rob ? κ66b 80tυ8 nυfjt m6,λ6 80f 4μ6ω 9tj
```

Checks & Balances



```
'true'           => true
'false'          => false
'42'             => 42
'2007-08-01T23:55:35.000Z' => Wed Aug 01 23:55:35 UTC 2007
'2007-08-01T23:22:32.000Z' => Wed Aug 01 23:22:32 UTC 2007
```

```
class CoercibleString < String
  attr_accessor :generator
  def coerce
    attempt = nil
    break unless (attempt = coercions.next).nil? while coercions.next?
    attempt.nil? ? self : attempt
  end

  private
  def coercions
    Generator.new do |self.generator|
      try { self == 'true' }
      try { [self == 'false', false] }
      try { Integer(self) }
      try { Date.parse(self) }
    end
  end
end

def try
  attempt, desired = yield
  generator.yield(desired.nil? ? attempt : desired) if attempt
rescue ArgumentError
  generator.yield nil
end
end
```

```
class CoercibleString < String
  def coerce
    case self
    when 'true':           true
    when 'false':         false
    when /^\d+$/:         Integer(self)
    when datetime_format: Time.parse(self)
    else
      self
    end
  end
end
```

60q

60q

60q

Thank You